Lab Number	5
Experiment Number	4
Experiment Title	Regression Analysis for Stock Prediction
Date of Experiment	23/01/2025
Date of Submission	30/01/2025

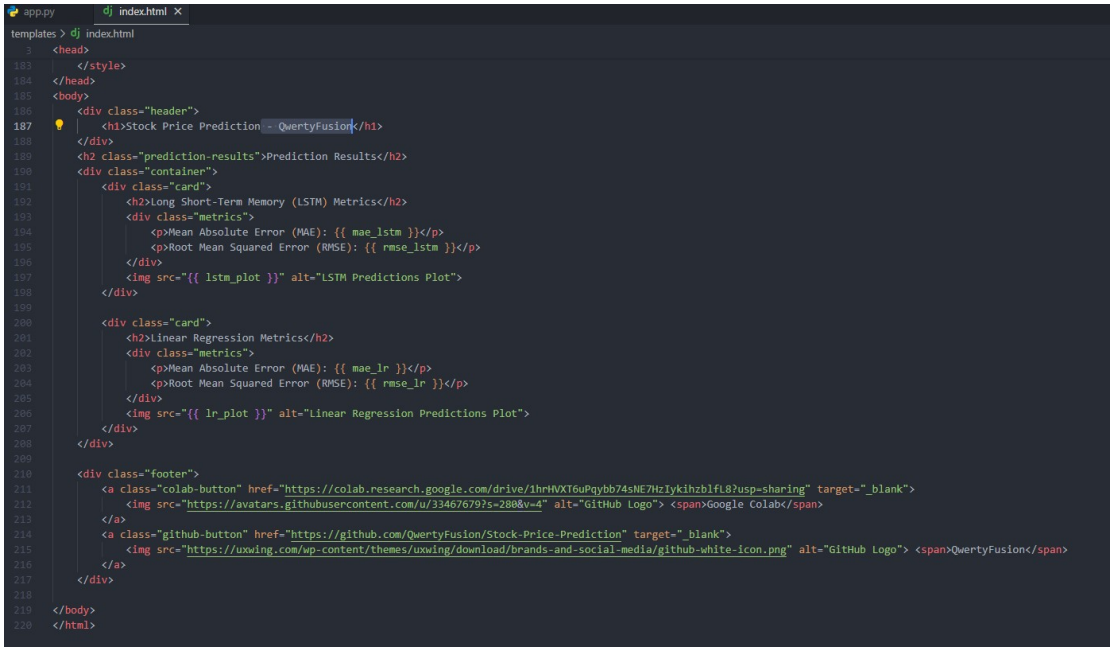
1. Objective:-

To perform stock price prediction using Linear Regression and LSTM models.

2. Procedure:- (Steps Followed)

1. Collect historical stock price data.
2. Preprocess the data for analysis (missing data, scaling, splitting into train/test).
3. Implement Linear Regression to predict future stock prices.
4. Design and train an LSTM model for time-series prediction.
5. Compare the accuracy of both models.
6. Create a Flask backend for model predictions.
7. Build a frontend to visualize predictions using charts and graphs.

3. Code:-



app.py index.html

templates > index.html

```
<head>
    </style>
</head>
<body>
    <div class="header">
        <h1>Stock Price Prediction - QwertyFusion</h1>
    </div>
    <h2 class="prediction-results">Prediction Results</h2>
    <div class="container">
        <div class="card">
            <h2>Long Short-Term Memory (LSTM) Metrics</h2>
            <div class="metrics">
                <p>Mean Absolute Error (MAE): {{ mae_lstm }}</p>
                <p>Root Mean Squared Error (RMSE): {{ rmse_lstm }}</p>
            </div>
            <img src="{{ lstm_plot }}" alt="LSTM Predictions Plot">
        </div>

        <div class="card">
            <h2>Linear Regression Metrics</h2>
            <div class="metrics">
                <p>Mean Absolute Error (MAE): {{ mae_lr }}</p>
                <p>Root Mean Squared Error (RMSE): {{ rmse_lr }}</p>
            </div>
            <img src="{{ lr_plot }}" alt="Linear Regression Predictions Plot">
        </div>
    </div>

    <div class="footer">
        <a class="colab-button" href="https://colab.research.google.com/drive/1hrHVXT6uPqybb74sNE7HzIykihzblfL8?usp=sharing" target="_blank">
            <img src="https://avatars.githubusercontent.com/u/33467679?s=280&v=4" alt="GitHub Logo"> <span>Google Colab</span>
        </a>
        <a class="github-button" href="https://github.com/QwertyFusion/Stock-Price-Prediction" target="_blank">
            <img src="https://uxwing.com/wp-content/themes/uxwing/download/brands-and-social-media/github-white-icon.png" alt="GitHub Logo"> <span>QwertyFusion</span>
        </a>
    </div>

</body>
</html>
```

app.py

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from flask import Flask, render_template
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
from keras.models import Sequential
from keras.layers import LSTM, Dense
import json
import os

app = Flask(__name__)

# Path to the directory where the downloaded zip file is placed.
DATA_DIR = "model_data"  # Make sure this folder exists.

@app.route('/')
def index():
    # Check if the zip file exists.  If not, instruct the user to upload it.
    zip_file_path = os.path.join(DATA_DIR, "model_outputs.zip")
    if not os.path.exists(zip_file_path):
        return "Please upload the `model_outputs.zip` file to the 'model_data' directory."

    # Extract the zip file (only if it hasn't been extracted already).
    extracted_flag_file = os.path.join(DATA_DIR, "extracted.flag") # Creating a flag to check if it has been already extracted.
    if not os.path.exists(extracted_flag_file):
        import zipfile
        with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
            zip_ref.extractall(DATA_DIR)
        # Create an empty file to indicate the extraction is complete.
        open(extracted_flag_file, "w").close()

    # Load the metrics with the info of the mae and rmse for linear regression and lstm
    metrics_path = os.path.join(DATA_DIR, 'metrics.json')
    with open(metrics_path, 'r') as f:
        metrics = json.load(f)

    mae_lr = metrics['mae_lr']
    rmse_lr = metrics['rmse_lr']
    mae_lstm = metrics['mae_lstm']
    rmse_lstm = metrics['rmse_lstm']

    # Render the template
    return render_template('index.html',
                           lstm_plot='static/lstm_predictions.png', # Downloaded image should be placed in the static folder
                           lr_plot='static/lr_predictions.png', # Downloaded image should be placed in the static folder
                           mae_lr=mae_lr,
                           rmse_lr=rmse_lr,
                           mae_lstm=mae_lstm,
                           rmse_lstm=rmse_lstm)


if __name__ == '__main__':
    # Create the data directory if it doesn't exist
    os.makedirs(DATA_DIR, exist_ok=True)
    app.run(debug=True, port=8151) # Custom port based on my roll number (2205151), default port 5000 was busy.
```

AD Lab 5 Exp 4.ipynb

File Edit View Insert Runtime Tools Help

Share

+ Code + Text

Gemini

Requirement already satisfied: markdown-it-py>=2.2.0->rich->keras) (0.1.2)

Step 1: Data Preparation

```python
# Step 1: Data Preparation
import pandas as pd
import numpy as np

# Load the dataset
df = pd.read_csv('GOOG.csv')  # Replace with your actual file name

# Features: Using 'open', 'high', 'low', and 'volume'
X = df[['open', 'high', 'low', 'volume']].values

# Target: Predicting the 'close' price
y = df['close'].values
```

+ Code + Text

Step 2: Model Training

```python
# Step 2: Model Training
from keras.models import Sequential
from keras.layers import LSTM, Dense
from sklearn.linear_model import LinearRegression

# Reshape data for LSTM
X_lstm = X.reshape((X.shape[0], X.shape[1], 1))

# Create LSTM model
model_lstm = Sequential()
model_lstm.add(LSTM(50, activation='relu', input_shape=(X_lstm.shape[1], 1)))
model_lstm.add(Dense(1))
model_lstm.compile(optimizer='adam', loss='mse')

# Train the model
model_lstm.fit(X_lstm, y, epochs=200, verbose=0)
pred_lstm = model_lstm.predict(X_lstm)
```

AD Lab 5 Exp 4.ipynb

File Edit View Insert Runtime Tools Help

Share

+ Code + Text

Gemini

Step 3: Generate Predictions and Calculate Metrics

```python
[32] # Step 3: Generate Predictions and Calculate Metrics
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Calculate metrics for Linear Regression
mae_lr = mean_absolute_error(y, pred_lr)
mse_lr = mean_squared_error(y, pred_lr)
rmse_lr = np.sqrt(mse_lr)

# For LSTM, you can calculate similar metrics
mae_lstm = mean_absolute_error(y, pred_lstm)
mse_lstm = mean_squared_error(y, pred_lstm)
rmse_lstm = np.sqrt(mse_lstm)
```

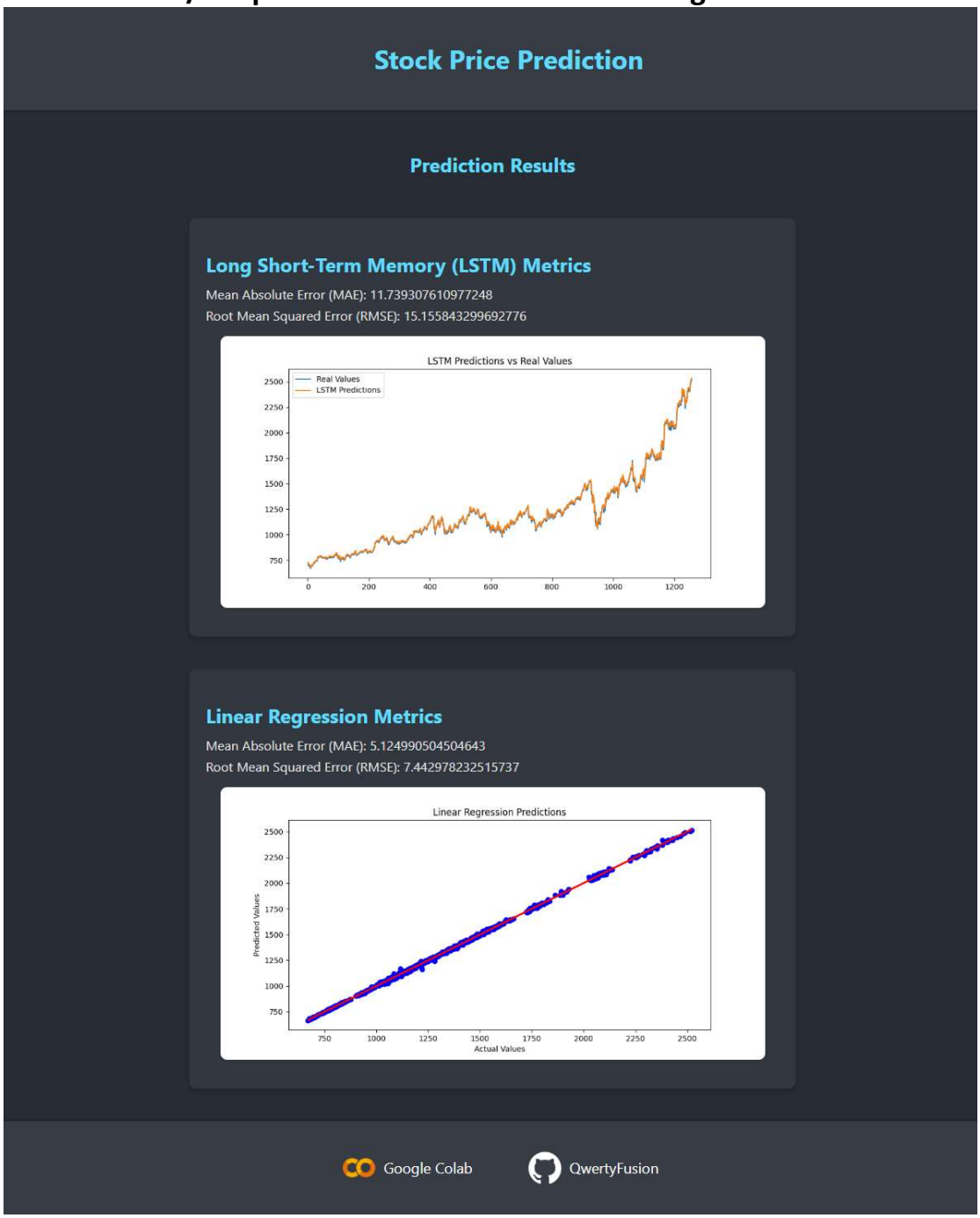
Step 4: Plotting

+ Code + Text

```python
[33] # Step 4: Plotting
import matplotlib.pyplot as plt

# Plot LSTM predictions
plt.figure(figsize=(10, 5))
plt.plot(y, label='Real Values')
plt.plot(pred_lstm, label='LSTM Predictions')
plt.title('LSTM Predictions vs Real Values')
plt.legend()
plt.savefig('lstm_predictions.png')  # Save the plot as an image
```

4. Results/Output:- Entire Screen Shot including Date & Time



5. Remarks:-

Signature of the Student

Signature of the Lab Coordinator

_____ _____

(Name of the Student) (Name of the Coordinator)